

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

Method and Apparatus for Event Handling In an Enterprise

Inventors:

Ashvinkumar J. Sanghvi

Howard M. Hance

Lev Novik

Fred E. Shaudys

ATTORNEY'S DOCKET NO. MS1-693US

0932508-060201

1 **RELATED APPLICATIONS**

2 This application claims the benefit of U.S. Provisional Application No.
3 60/210,347, filed June 7, 2000.

4
5 **TECHNICAL FIELD**

6 The present invention relates to computing systems and, more particularly,
7 to the handling of events generated by components, services and applications in a
8 computing environment.

9
10 **BACKGROUND**

11 Computer systems, such as servers and desktop personal computers, are
12 expected to operate without constant monitoring. These computer systems
13 typically perform various tasks without the user's knowledge. When performing
14 these tasks, the computer system often encounters events that require a particular
15 action (such as logging the event, generating an alert for a particular system or
16 application, or performing an action in response to the event). Various
17 mechanisms are available to handle these events.

18 A computing enterprise typically includes one or more networks, services,
19 and systems that exchange data and other information with one another. The
20 enterprise may include one or more security mechanisms to safeguard data and
21 authenticate users and may utilize one or more different data transmission
22 protocols. At any particular time, one or more networks, services or systems may
23 be down (e.g., powered down or disconnected from one or more networks).
24 Networks, services or systems can be down for scheduled maintenance, upgrades,
25

1 overload or failure. Application programs attempting to obtain event data must
2 contend with the various networks, services, and systems in the enterprise when
3 they are down. Additionally, application programs must contend with the security
4 and network topology limitations of the enterprise as well as the various protocols
5 used in the enterprise.

6 Existing operating system components, services, and applications generate
7 events having a variety of different formats. Typically, the events are stored in
8 different files or databases (e.g., a file or database on the local system). These
9 stored events are accessed via different application programs using different
10 application programming interfaces (APIs). Thus, to retrieve event data in this
11 type of system, an application program must know where to locate the stored event
12 data and how to read or "decode" the particular event data. Each time a new type
13 of event (e.g., having a new storage location, a new format, and/or a new API) is
14 introduced, each application program that desires the new event data must be
15 adapted to locate and retrieve the new event data.

16 The systems and methods described herein address these limitations by
17 providing a centralized mechanism for collecting and storing event data. The
18 systems and methods also provide a uniform event-handling process and
19 infrastructure.

20 21 22 SUMMARY

23 The event-handling system and method described herein provide a
24 centralized architecture and procedure for managing event data. The centralized
25

1 handling of event data uses a common interface to access event data, regardless of
2 the event source, event data format, network topology or security mechanisms
3 contained in the enterprise. Additional event sources can be added to the
4 enterprise without requiring changes to the event-handling system. The event-
5 handling system is also capable of grouping together multiple devices and
6 assigning a common policy to all devices in the group.

7 In one embodiment, multiple devices are assigned to a group. At least one
8 event-handling policy is assigned to the group such that the assigned policy is
9 associated with each of the multiple devices in the group. A current state of each
10 device is evaluated before the assigned policy is applied to the device.

11 In a described embodiment, a particular device may be assigned to two or
12 more groups.

13 In a particular embodiment, the event-handling policy identifies the types
14 of events that are provided to each device.

15 16 17 **BRIEF DESCRIPTION OF THE DRAWINGS**

18 Fig. 1 illustrates a block diagram of a system that receives event
19 information from multiple event providers and provides event information to
20 multiple event consumers.

21 Fig. 2 illustrates a block diagram of a system that receives events and logs
22 those events to an event log.

23 Fig. 3 is a flow diagram illustrating an event-handling procedure.

24 Fig. 4 illustrates an environment in which multiple systems are arranged
25 into three different groupings, all of which are coupled to an event log.

1 Fig. 5 is a flow diagram illustrating a procedure for establishing and
2 managing groupings of computer systems.

3 Fig. 6 illustrates the combining of multiple policies into a single merged
4 policy set.

5 Fig. 7 is a flow diagram illustrating a procedure for generating a merged
6 policy set from multiple policies.

7 Fig. 8 illustrates an arrangement of multiple policy templates having
8 partially overlapping policy elements.

9 Fig. 9 illustrates an example of a suitable operating environment in which
10 the event-handling system and method may be implemented.

11 12 13 **DETAILED DESCRIPTION**

14 The systems and methods described herein provide for the centralized
15 handling of event data generated by various event sources in an enterprise. The
16 use of a common data format, a centralized event data storage device, and a
17 common interface to obtain event data improves access to the event data and
18 reduces administrative tasks associated with the handling of event data generated
19 throughout the enterprise. The same event data format is used regardless of the
20 source of the event data (also referred to as an event provider) or the users of the
21 event data (also referred to as an event consumer). As the systems, applications,
22 and topology of the enterprise changes, the event data format remains unchanged.

23 Web-Based Enterprise Management (WBEM) provides uniform access to
24 management information throughout an enterprise. WBEM is an industry
25

1 initiative to develop technology for accessing management information in an
2 enterprise environment. This management information includes, for example,
3 information on the state of system memory, inventories of currently installed client
4 applications, and other information related to the status of the system. A particular
5 embodiment of the event-handling system is implemented using Windows
6 Management Instrumentation (WMI) developed by Microsoft Corporation of
7 Redmond, Washington, which provides an infrastructure to handle various events
8 generated by event sources throughout an enterprise. WMI is Microsoft
9 Corporation's implementation of WBEM.

10 WMI technology enables systems, applications, networks, and other
11 managed components to be represented using the Common Information Model
12 (CIM) designed by the Distributed Management Task Force (DMTF). CIM is an
13 extensible data model for representing objects that exist in typical management
14 environments. CIM is able to model anything in the managed environment,
15 regardless of the location of the data source. The Managed Object Format (MOF)
16 language is used to define and store modeled data. In addition to data modeling,
17 WMI provides a set of base services that include query-based information retrieval
18 and event notification. Access to these services and to the management data is
19 provided through a single programming interface.

20 WMI classes define the basic units of management. Each WMI class is a
21 template for a type of managed object. For example, Win32_DiskDrive is a model
22 representing a physical disk drive. For each physical disk drive that exists, there is
23 an instance of the Win32_DiskDrive class. WMI classes may contain properties,
24 which describe the data of the class and methods, which describe the behavior of
25

1 the class.

2 WMI classes describe managed objects that are independent of a particular
3 implementation or technology. WMI includes an eventing subsystem that follows
4 the publish-subscribe model, in which an event consumer subscribes for a
5 selection of events (generated by one or more event providers) and performs an
6 action as a result of receiving the event. WMI also provides a centralized
7 mechanism for collecting and storing event data. This stored event data is
8 accessible by other systems via WMI tools and/or application programming
9 interfaces (APIs).

10 Although particular embodiments are discussed herein as using WMI,
11 alternate embodiments may utilize any enterprise management system or
12 application, whether web-based or otherwise. The event providers and event
13 consumers discussed herein are selected for purposes of explanation. The
14 teachings of the present invention can be used with any type of event provider and
15 any type of event consumer. Additionally, the event-handling system and method
16 described herein can be applied to any type of enterprise or other arrangement of
17 computing devices, applications, and/or networks.

18
19 Fig. 1 illustrates a block diagram of a system 100 that receives event
20 information from multiple event providers 108 (i.e., event sources) and provides
21 event information to multiple event consumers 102 (i.e., the users of the event
22 data). System 100 includes a WMI module 106, which receives event data from
23 multiple event sources 108 and receives requests for information (e.g., notification
24 of particular events) from multiple event consumers 102. Event sources 108 may
25 include, for example, managed nodes or managed systems in a network. The

multiple event sources are identified as event providers 110. The multiple event consumers are identified as applications 104.

WMI module 106 shown in Fig. 1 represents the managed node layer of the WMI module. As discussed below, the WMI module 106 may also include a central store layer, which may include user interface functionality. The different layers of WMI module 106 manage different types of activities and/or perform different types of functions.

Event providers 110 include, for example, systems, services or applications that generate event data. An exemplary event provider is a disk drive (or an application that monitors the status of a disk drive). The disk drive may generate an event indicating the available storage capacity on the disk drive or indicating the amount of data currently stored on the disk drive. The disk drive may also generate an event indicating that the disk drive is nearly full of data (e.g., when ninety-five percent or more of the disk drive's capacity is used).

Event consumers 102 may request to be notified of certain events (also referred to as "subscribing" to an event). An example event consumer is an application that manages multiple storage devices in an enterprise. The application may request to receive events generated by any of the disk drives or other storage devices in the enterprise. The application can use this event information to distribute storage tasks among the multiple storage devices based on the available capacity of each device and/or the quantity of read or write requests received by each storage device.

Fig. 2 illustrates a block diagram of a system 150 that receives events and logs those events to an event log. System 150 includes a central store layer of

1 WMI module 106, which is coupled to multiple user interface (UI) applications
2 152. UI applications 152 are used to access WMI module 106 to retrieve data,
3 manage systems, and configure various enterprise management parameters. The
4 central store layer of WMI module 106 provides for the centralized logging and
5 storage of event data received from various nodes and various networks in an
6 enterprise. WMI module 106 is also coupled to receive events 162 from one or
7 more event sources. For example, events may be received from the managed node
8 layer of WMI module 106, discussed above with respect to Fig. 1, from an event
9 forwarding application (e.g., application 104), or from one or more event
10 providers (e.g., event provider 110).

11 System 150 also includes a set of policies 160, which are accessible by
12 WMI module 106. Policies 160 may control the configuration of one or more
13 systems in the enterprise. Other policies may define various activities, such as
14 event filtering, event correlation, and the forwarding of events to particular
15 devices or applications. A database 156 is coupled to WMI module 106.
16 Database 156 stores various information related to the enterprise. For example,
17 database 156 can store event data (i.e., creating an event log), policy data, and
18 enterprise configuration information.

19 WMI module 106 is also coupled to an event log 158. The event log 158
20 uses WMI features to provide a distributed architecture that is capable of selecting,
21 filtering, correlating, forwarding, storing, and delivering event data in an
22 enterprise. The event log 158 allows users, such as administrators, to request data
23 related to a particular event, request data from a particular node or device in the
24 enterprise, define the manner in which events are correlated with one another,
25 define how certain events should be forwarded, and define how to store event data.

TABLE 1

Event Provider	Description of Events Provided
Win32 Provider	Supplies information about the operating system, computer system, peripheral devices, file systems, and security.
WDM Provider	Supplies low-level Windows Driver Model (WDM) information for user input devices, storage devices, network interfaces, and communications ports.
Event Log Provider	Allows the reading of Windows NT event log entries, controls the configuration of event log administrative options, and event log backup.
Registry Provider	Allows registry keys to be created, read, and written. WMI events can be generated when specified Registry keys are modified.
Performance Counter Provider	Exposes the raw performance counter information used to compute various performance values.
Active Directory Provider	Acts as a gateway to information stored in Microsoft Active Directory services. Allows information from both WMI and Active Directory to be accessed using a single API.
Windows Installer Provider	Supplies information about applications installed with the Windows Installer.
SNMP Provider	Acts as a gateway to systems and devices that use SNMP for management. Allows SNMP traps to be automatically mapped to WMI events.

Fig. 3 is a flow diagram illustrating an event-handling procedure 200. The WMI module monitors event activity throughout the enterprise (block 202). The procedure 200 determines whether event data has been received from an event

1 provider (block 204). If event data has been received, the WMI module records
2 the event data and initiates any appropriate actions (block 206). An appropriate
3 action includes notifying an event consumer of the event (e.g., if the event
4 consumer previously subscribed to such an event).

5 At block 208, the procedure 200 determines whether a new subscription for
6 event information has been received. The procedure 200 may also determine
7 whether a request to revise an existing subscription has been received. If a new
8 subscription (or a revised subscription) is received, the procedure continues to
9 block 210 where the WMI module retrieves the requested event information and
10 provides the information to the requesting event customer. Alternatively, the
11 procedure may log the subscription request and notify the requesting event
12 consumer when the next event is received that qualifies under the consumer's
13 subscription request.

14 The WMI module allows multiple systems in an enterprise to be grouped
15 together such that various event policies are assigned to the group of systems,
16 rather than assigning the same set of policies to each individual system. This
17 grouping of systems simplifies the administrative task of assigning event policies
18 to systems within the enterprise.

19 Fig. 4 illustrates an environment 300 in which multiple systems are
20 arranged into three different groupings, all of which are coupled to an event log.
21 The grouping of systems is rule-based depending both on the organization of the
22 enterprise and the properties of the various systems in the enterprise. Additionally,
23 the grouping of systems may be based on the current state and configuration of the
24 systems in the enterprise. Example groups may include all computer systems
25 running a particular version of an operating system, all systems located in a

1 particular geographic region (e.g., Europe), and all systems that have more than
2 500 Megabytes of free disk space.

3 The rule-based grouping of systems simplifies the administrative tasks by
4 not requiring the manual maintenance of lists identifying the current configuration
5 and current state of each system in the enterprise. The current state of each system
6 is evaluated before each policy is applied, thereby reducing the likelihood that
7 previously determined state information is no longer valid. As systems enter and
8 leave the enterprise or change configuration, the correct policies are applied to the
9 systems regardless of these ongoing changes to the enterprise.

10 Environment 300 in Fig. 4 includes three separate groupings 302, 304, and
11 306. Each grouping 302-306 is coupled to event log 308, which maintains and
12 evaluates the state and configuration information of the systems in environment
13 300. Group 302 includes five systems, group 304 includes two systems, and group
14 306 includes two systems, one of which is also included in group 302. Thus, a
15 particular system may be located in two or more groups.

16 As mentioned above, the grouping of systems is used to simplify the
17 assignment of policies by assigning similar policies to the group instead of
18 assigning the same policies to each individual system. Additionally, this grouping
19 of systems simplifies the management of the system by allowing the administrator
20 to work with fewer groups instead of a larger number of individual systems, many
21 of which have redundant policies.

22 Fig. 5 is a flow diagram illustrating a procedure 400 for establishing and
23 managing groupings of computer systems. Initially, the administrator of the
24 enterprise (or a portion of the enterprise) defines one or more groups (block 402).
25 Next, the administrator assigns policies to each defined group (block 404). Each

005090-8525286
1098757-060501

1 system in the group becomes associated with the assigned policies, in the same
2 manner as if the policies were separately assigned to each of the individual
3 systems. Block 406 determines whether a particular system is a member of one or
4 more groups. In one implementation, this determination is performed at policy-
5 evaluation time (i.e., when policies are applied to one or more systems in the
6 enterprise). A particular system may be a member of a group at one instance, but
7 not a member of the same group at a different time. For example, if a group
8 includes all systems that have a modem installed, a system that was previously a
9 member of the group will not be a member if the modem is removed from the
10 system. Since the group membership is determined at policy-evaluation time, a
11 particular system may be removed from a group without any action on the part of
12 the administrator or other user.

13 At block 408, the procedure 400 determines whether a new group has been
14 defined. If a new group has been defined, then the administrator assigns policies
15 to the new group (block 410). The procedure then returns to block 406 to
16 determine whether a particular system is a member of the new group as well as
17 other existing groups. Administrators (or other users) may generate new policies
18 and apply those new policies to particular systems and/or groups of systems. New
19 policies that are applied to a group are automatically applied to all systems in the
20 group.

21 In a particular embodiment, systems in an enterprise are grouped according
22 to the department or organization division with which they are associated. For
23 example, one group of systems may be associated with the production department,
24 another group associated with the marketing department, and a third group
25 associated with the customer service department. Each department may have

1 different needs with respect to their policies, but the systems within a particular
2 department are likely to have many policies in common. For example, an
3 accounting department may have stricter security requirements and, therefore,
4 require a different set of policies.

5 Fig. 6 illustrates the combining of multiple policies 502, 504, 506, and 508
6 into a single merged policy set 510. Event log 512 communicates with the merged
7 policy set 510. This merging of policies allows several policies to be merged
8 together into a single policy. In a particular implementation, policies are applied
9 by administrators and stored in a central location. The appropriate policies for a
10 particular system are selected, ordered, merged and applied by the WMI module.
11 When the policy is applied, the desired event filters and bindings are created at the
12 appropriate systems throughout the enterprise.

13 The policy elements that are complementary with one another are appended
14 to the new, merged policy set 510. If two or more policy elements are in conflict
15 with one another, then the conflict is resolved by applying a conflict-resolution
16 algorithm, discussed below.

17 Fig. 7 is a flow diagram illustrating a procedure 600 for generating a
18 merged policy set from multiple policies. Initially, the procedure 600 identifies
19 multiple policies to be merged together and compares related policies individually
20 (block 602). Next, the procedure determines whether the policies being compared
21 are in conflict with one another (block 604). If the policies are not in conflict with
22 one another, the non-conflicting policies are added to the merged policy set (block
23 606). However, if the policies are conflicting, the procedure continues to block
24 608 where the conflicting policy templates are arranged in order from global
25 policies to local policies. The procedure 600 then determines the intersection of

1 the multiple policy templates (block 610) and selects a preferred policy template
2 (block 612). If additional policies remain to be evaluated, the procedure returns to
3 block 604. Otherwise, the procedure ends after evaluating all policies.

4 Fig. 8 illustrates an arrangement 700 of multiple policy templates having
5 partially overlapping policy elements. Each policy template includes multiple
6 properties. One property represents an "allowable range" for the policy and
7 another property represents a "preferred value" for the policy. These property
8 values will affect the outcome of the application of the policy, which, in turn,
9 causes the creation of event filters, bindings, and other activities to apply the
10 policy throughout the enterprise.

11 The arrangement 700 is used to eliminate conflicts between multiple
12 policies being merged into a single policy set. As mentioned above with respect to
13 block 608 in Fig. 7, the policy templates are arranged from global policy templates
14 (e.g., policies that define the broad configuration and operation objectives for the
15 entire enterprise) at the top of Fig. 8 to local policy templates (e.g., policies that
16 are specific to a particular device or application) at the bottom of Fig. 8. For
17 example, a global policy template 702 may be created or defined by one or more
18 administrators that are responsible for administering the entire enterprise. A local
19 policy template 712 may be created or defined by an administrator that is
20 responsible for a particular portion of an enterprise, such as a particular group of
21 systems or systems in a specific location. Additional policy templates 704, 706,
22 708, and 710 each contain varying levels of policies ranging from nearly global
23 policies (policy template 704) to nearly local policies (policy template 710).

24 After the policy templates are arranged as shown in Fig. 8, it is necessary to
25 find the intersection of all policy templates. In the example of Fig. 8, the

1 intersection of five of the policy templates is shown by the two broken lines 714
2 and 716. This intersection of five policy templates defines an "allowed range" that
3 satisfies the majority of policy templates. Note that policy template 708 is
4 discarded because the policies are in conflict with (e.g., opposed to) the policy
5 range defined by the intersection of the other five policy templates 702, 704, 706,
6 710, and 712. Within each policy template 702, 704, 706, 708, 710 and 712, is a
7 preferred range or preferred value 720 (identified by a "P" surrounded by a box)
8 associated with the policy template.

9 Finally, a "preferred range" is selected. The preferred range (or preferred
10 policy) has all properties set to preferred properties. Each preferred property is a
11 preferred policy from the policy closest to the system (i.e., the bottom of Fig. 8)
12 and still within the "allowed range". In the example of Fig. 8, the preferred range
13 for the merged policy template is the preferred range associated with policy
14 template 710, because it is the preferred range closest to the system that is also
15 within the allowed range. The preferred range associated with policy template 712
16 is not selected because that preferred range is not within the allowed range.

17 The conflict resolution procedure discussed above achieves customization
18 of the policies on a particular system based on the preferences of the administrator
19 closest to the system (i.e., the administrator most knowledgeable about the system
20 and responsible for the system) while staying within the policy ranges dictated by
21 all administrators with a higher level of authority.

22 The following are example policy templates:
23
24
25

Policy template 1:

Policy type: policy forwarding

Destination range: Sys-red, Sys-blue, Sys-green

Destination preferred: Sys-blue

Policy template 2:

Policy type: policy forwarding

Destination range: Sys-red, Sys-blue

Destination preferred: Sys-red

In this example, Policy template 1 is set at a global level and Policy template 2 is set at a local level. When the two policy templates are merged, the resulting merged template is:

Policy template Merged:

Policy type: policy forwarding

Destination range: Sys-red, Sys-blue, Sys-green

Destination preferred: Sys-red

Thus, the preferred range or value is selected from the lowest (most local) possible level. In this case, the preferred range of the merged policy template is the preferred range of the local policy template (Sys-red).

Fig. 9 illustrates an example of a suitable operating environment in which the event handling mechanism described herein may be implemented. The illustrated operating environment is only one example of a suitable operating environment and is not intended to suggest any limitation as to the scope of use or functionality of the invention. Other well-known computing systems, environments, and/or configurations that may be suitable for use with the invention include, but are not limited to, personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based

1 systems, programmable consumer electronics, gaming consoles, cellular
2 telephones, network PCs, minicomputers, mainframe computers, distributed
3 computing environments that include any of the above systems or devices, and the
4 like.

5 Fig. 9 shows a general example of a computer 842 that can be used in
6 accordance with the invention. Computer 842 is shown as an example of a
7 computer that can perform the various functions described herein. Computer 842
8 includes one or more processors or processing units 844, a system memory 846,
9 and a bus 848 that couples various system components including the system
10 memory 846 to processors 844.

11 The bus 848 represents one or more of any of several types of bus
12 structures, including a memory bus or memory controller, a peripheral bus, an
13 accelerated graphics port, and a processor or local bus using any of a variety of
14 bus architectures. The system memory 846 includes read only memory (ROM)
15 850 and random access memory (RAM) 852. A basic input/output system (BIOS)
16 854, containing the basic routines that help to transfer information between
17 elements within computer 842, such as during start-up, is stored in ROM 850.
18 Computer 842 further includes a hard disk drive 856 for reading from and writing
19 to a hard disk, not shown, connected to bus 848 via a hard disk drive interface 857
20 (e.g., a SCSI, ATA, or other type of interface); a magnetic disk drive 858 for
21 reading from and writing to a removable magnetic disk 860, connected to bus 848
22 via a magnetic disk drive interface 861; and an optical disk drive 862 for reading
23 from and/or writing to a removable optical disk 864 such as a CD ROM, DVD, or
24 other optical media, connected to bus 848 via an optical drive interface 865. The
25 drives and their associated computer-readable media provide nonvolatile storage

1 of computer readable instructions, data structures, program modules and other data
2 for computer 842. Although the exemplary environment described herein employs
3 a hard disk, a removable magnetic disk 860 and a removable optical disk 864, it
4 will be appreciated by those skilled in the art that other types of computer readable
5 media which can store data that is accessible by a computer, such as magnetic
6 cassettes, flash memory cards, random access memories (RAMs), read only
7 memories (ROM), and the like, may also be used in the exemplary operating
8 environment.

9 A number of program modules may be stored on the hard disk, magnetic
10 disk 860, optical disk 864, ROM 850, or RAM 852, including an operating system
11 870, one or more application programs 872, other program modules 874, and
12 program data 876. A user may enter commands and information into computer
13 842 through input devices such as keyboard 878 and pointing device 880. Other
14 input devices (not shown) may include a microphone, joystick, game pad, satellite
15 dish, scanner, or the like. These and other input devices are connected to the
16 processing unit 844 through an interface 868 that is coupled to the system bus
17 (e.g., a serial port interface, a parallel port interface, a universal serial bus (USB)
18 interface, etc.). A monitor 884 or other type of display device is also connected to
19 the system bus 848 via an interface, such as a video adapter 886. In addition to the
20 monitor, personal computers typically include other peripheral output devices (not
21 shown) such as speakers and printers.

22 Computer 842 operates in a networked environment using logical
23 connections to one or more remote computers, such as a remote computer 888.
24 The remote computer 888 may be another personal computer, a server, a router, a
25 network PC, a peer device or other common network node, and typically includes

1 information such as computer readable instructions, data structures, program
2 modules or other data. Computer storage media includes, but is not limited to,
3 RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM,
4 digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic
5 tape, magnetic disk storage or other magnetic storage devices, or any other media
6 which can be used to store the desired information and which can be accessed by
7 computer 842. Communication media typically embodies computer readable
8 instructions, data structures, program modules or other data in a modulated data
9 signal such as a carrier wave or other transport mechanism and includes any
10 information delivery media. The term "modulated data signal" means a signal that
11 has one or more of its characteristics set or changed in such a manner as to encode
12 information in the signal. By way of example, and not limitation, communication
13 media includes wired media such as wired network or direct-wired connection,
14 and wireless media such as acoustic, RF, infrared and other wireless media.
15 Combinations of any of the above should also be included within the scope of
16 computer readable media.

17 The invention has been described in part in the general context of
18 computer-executable instructions, such as program modules, executed by one or
19 more computers or other devices. Generally, program modules include routines,
20 programs, objects, components, data structures, etc. that perform particular tasks
21 or implement particular abstract data types. Typically the functionality of the
22 program modules may be combined or distributed as desired in various
23 embodiments.

24 For purposes of illustration, programs and other executable program
25 components such as the operating system are illustrated herein as discrete blocks,

1 although it is recognized that such programs and components reside at various
2 times in different storage components of the computer, and are executed by the
3 data processor(s) of the computer.

4 Although the description above uses language that is specific to structural
5 features and/or methodological acts, it is to be understood that the invention
6 defined in the appended claims is not limited to the specific features or acts
7 described. Rather, the specific features and acts are disclosed as exemplary forms
8 of implementing the invention.

10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
T05090" 86254360